Tangent Fisher Vector on Matrix Manifolds for Action Recognition

Guan Luo, *Member, IEEE*, Jiutong Wei, Weiming Hu, *Senior Member, IEEE*, and Stephen J. Maybank, *Fellow, IEEE*

Abstract—In this paper, we address the problem of representing and recognizing human actions from videos on matrix manifolds. For this purpose, we propose a new vector representation method, named tangent Fisher vector, to describe video sequences in the Fisher kernel framework. We first extract dense curved spatio-temporal cuboids from each video sequence. Compared with the traditional 'straight cuboids', the dense curved spatio-temporal cuboids contain much more local motion information. Each cuboid is then described using a linear dynamical system (LDS) to simultaneously capture the local appearance and dynamics. Furthermore, a simple yet efficient algorithm is proposed to learn the LDS parameters and approximate the observability matrix at the same time. Each video sequence is thus represented by a set of LDSs. Considering that each LDS can be viewed as a point in a Grassmann manifold, we propose to learn an intrinsic GMM on the manifold to cluster the LDS points. Finally a tangent Fisher vector is computed by first accumulating all the tangent vectors in each Gaussian component, and then concatenating the normalized results across all the Gaussian components. A kernel is defined to measure the similarity between tangent Fisher vectors for classification and recognition of a video sequence. This approach is evaluated on the state-of-the-art human action benchmark datasets. The recognition performance is competitive when compared with current state-of-the-art results.

Index Terms—Action recognition, Fisher vector, Grassmann manifold, Hankel matrix, matrix manifold.

1 Introduction

Modeling and recognizing human activities from videos is a key component in many promising applications including visual surveillance, human computer interaction, and video summarization. In the past couple of decades, many papers on human activity analysis have been published. The surveys by Aggarwal and Ryoo [1] and Weinland *et al.* [2] provide a broad overview of these efforts. However, the challenges of understanding human actions still remain, due in part to the diversity of human movements and the variations of dynamic environments.

In much recent work, dynamical system methods [3], [4], [5], [6] are used to model the human movements and activities in video. In particular, linear dynamical systems (LDSs) are used widely because of their simplicity and efficiency. In addition, LDSs have the merit of capturing both action appearance and dynamics by decoupling video sequences into subspace poses and latent dynamics. Recent advances in system identification theory for measuring the distance between LDSs [7], [8], [9], [10] have made LDSs even more successful for the classification of high-dimensional time-series data. State-of-the-art results are achieved in applications ranging from classifying non-rigid dynamic textures [11], [12] to recognizing highly articulated human actions [3], [6], [13], [14].

On modeling a motion sequence with LDSs, the model parameters or the associated infinite observability matrix are generally used descriptors. However, these descriptors do live in a matrix manifold rather than a Euclidean vector space. In order to compare two LDSs, a distance or kernel metric needs to be defined by the system methods [7], [8], [9], or using geodesics on the matrix manifold [6], [15], [16]. Once a distance metric is defined, one way to achieve action recognition is to model the whole video sequence as an LDS [3], [14]. Then classifiers such as KNNs or SVMs are used to categorize a query video based on the pairwise distance matrix of the training videos. However, for complex human actions, it is no longer reasonable to model the whole video sequence with only one LDS. This is because complex human actions usually contain many spatio-temporal nonlinearities. One way to address this issue is to use nonlinear dynamical systems (NLDSs) [4], [5], [9], [17]. However, NLDSs usually require large amounts of training data to learn the model parameters. This makes it hard to balance the model accuracy and generalization. In addition, measuring distances between NLDSs is much more difficult than LDSs.

In this paper, we develop a new action representation method, named tangent Fisher vector (TFV), to combine the strengths of the dense local spatio-temporal features [18] and the Fisher kernel framework [19]. The main motivation behind this method has two aspects. First, we model each video sequence using a set of LDSs other than only one LDS or a NLDS, by extracting

- *G. Luo, J. Wei and W. Hu are with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, No. 95, Zhongguancun East Road, PO Box 2728, Beijing, 100190, P.R. China. E-mail: {gluo, wmhu}@nlpr.ia.ac.cn.*
- *S. J. Maybank is with the Department of Computer Science and Information Systems, Birkbeck College, Malet Street, London WC1E 7HX, United Kingdom. E-mail: sjmaybank@dcs.bbk.ac.uk.*

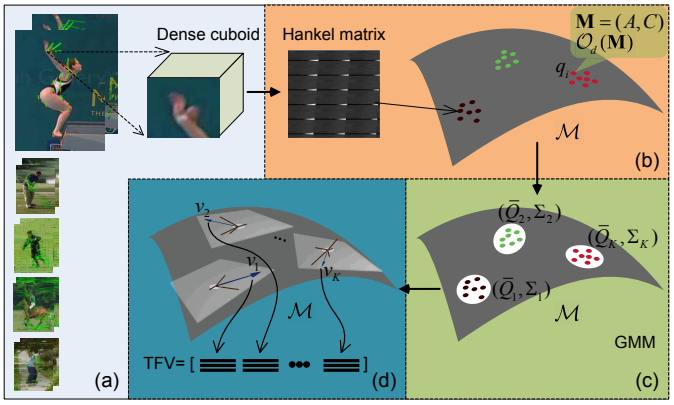


Fig. 1. System diagram of the proposed method. The system is composed of four main parts: (a) constructing dense cuboids, (b) learning observability matrices of LDSs, (c) fitting an intrinsic GMM, and (d) computing the tangent Fisher vector.

dense local spatio-temporal cuboids and describing each cuboid with an LDS. The spatio-temporal cuboids are generally small in size in both the spatial and temporal domains, thus the inherent dynamics in each spatio-temporal cuboid can be well approximated by an LDS. In this way, the action recognition problem reduces to classifying a set of LDSs. Second, considering that the LDSs are specified by points in a non-Euclidean matrix manifold, we extend the traditional Fisher kernel framework and propose an intrinsic GMM to aggregate the set of LDSs of a video sequence into a tangent Fisher vector (TFV). We demonstrate that the combination of dense LDSs and TFVs achieves state-of-the-art results on very challenging public data sets.

Fig. 1 gives an overview of the proposed method, which is composed of four main parts: constructing dense cuboids, learning observability matrices of LDSs, fitting an intrinsic GMM, and computing the tangent Fisher vector. Given a video sequence, we first densely sample and track optical flow fields to form dense trajectories [18]. The curved spatio-temporal cuboids are constructed as the local space-time volumes along the trajectories. Then we use a Hankel matrix approach to efficiently compute the observability matrix of an LDS for each cuboid to capture the local appearance and dynamics. Next an intrinsic GMM on the matrix manifold is learned to cluster the observability matrices. Finally, each LDS is classified by a Gaussian component, by viewing the learned GMM as a codebook. A tangent Fisher vector is computed by aggregating the gradient vectors with respect to the Karcher mean and covariance in each Gaussian component. A trace kernel of two TFVs is used to measure the similarity between two video sequences for classification and recognition.

The main contributions of this paper are summarized as follows:

- We propose a TFV representation on a matrix manifold to describe video sequences. To the best of our knowledge, this is the first work which models non-Euclidean descriptors in the Fisher kernel framework.
- We propose an intrinsic GMM learning algorithm on the matrix manifold to construct the codebook for the observability matrices.
- We carry out extensive experiments on eight public action data sets. We evaluate the performance with respect to the LDS parameters, the cuboid size and the number of Gaussians in the GMM. We compare the TFV representation of the LDS descriptor with five baseline descriptors and four baseline clustering algorithms. The results demonstrate the significant advantages of our method.

The remainder of this paper is organized as follows: Section 2 reviews related work on the LDS, matrix manifolds and extensions of the Fisher vector framework. Section 3 introduces the LDS descriptor, namely the observability matrix and describes how the observability matrix is learned. Section 4 introduces the matrix manifold, and proposes an intrinsic GMM clustering algorithm on the manifold. Section 5 presents the computation of the tangent Fisher vector. Section 6 shows the experimental results, including comparisons with competing methods for learning human actions. Section 7 is a conclusion of the paper.

2 Related Work

Section 1 briefly reviewed the work on LDS descriptors and the Fisher vector representation in order to make clear the motivation for this paper. In this section, these methods are reviewed in detail in order to put the work into context.

The modeling of human movements by LDSs has been studied for a long time [20]. In early works, an LDS is learned using complex Bayesian modeling and inference. In more recent work, suboptimal LDS parameters are learned in closed form [11], and the similarity between LDSs is measured by defining distance metrics in the model space [7], [8], [9], [10]. Turaga *et al.* [21] segment a video sequence in the temporal dimension, and learn an LDS for each segment. They compute the pairwise distances among all the LDSs, and cluster them using normalized cuts. The video sequence is then modeled as a cascade of cluster labels, and repetitive patterns are mined using a regular expression grammar. The segments produced by Turaga *et al.* [21] are usually insufficient for motion analysis. This suggests the use of dense cuboids obtained by segmenting in both spatial and temporal dimensions. Bissacco *et al.* [3] model sequential data with a non-Gaussian LDS by assuming that the noise process of the LDS is temporally independent and white. They decompose the observation sequence into a deterministic process and a stochastic process, and design a suboptimal algorithm to estimate the model parameters. They define a novel kernel-based distance to classify human gaits by considering the dynamics,

initial conditions, and input distribution. Considering that the non-Gaussian model is very difficult to identify and that describing a video sequence using only one linear model is not reasonable for most complex videos, we extract a large number of local cuboids in each video and describe them using linear Gaussian models. Li *et al.* [22] reformulate the observation sequence as a Hankel matrix, and model it as the output of an LDS. They do not estimate the LDS parameters, but exploit the subspace spanned by the columns of the Hankel matrix. They use the discriminant canonical correlations (DCC) between two subspaces to measure the distance of two video sequences for action recognition. We follow their ideas by further exploring the geometry of the subspace of the Hankel matrix, and represent LDSs as points in a matrix manifold.

The characterization of human actions using matrix manifolds has been a recent focus in the literature. Veeraraghavan *et al.* [23] model human silhouette shapes on a spherical manifold. A shape sequence is first normalized using dynamic time warping (DTW). Then a tangent space at the Procrustes mean shape is constructed, and an ARMA model is exploited to describe the shape sequence. The model distance is employed to compare different sequences for gait recognition. Considering that the silhouette or shape features are difficult to obtain on unconstrained human action data sets, most recent methods use the raw pixel values. Turaga *et al.* [6] describe the video sequence using ARMA model parameters. They compute the observability matrix, and interpret it as a point in a Grassmann manifold. They investigate the statistical modeling and Procrustes representation on the manifold, and learn class conditional densities and intrinsic K-means for classification and clustering on the tangent space of the Grassmann manifold. Lui *et al.* [16] use a modified high-order singular value decomposition (HOSVD) on the video sequence to obtain three factor matrices to capture the appearance, horizontal motion, and vertical motion respectively. These three factor matrices are points on a special matrix manifolds. They compute the distances on three tangent spaces, and put them together on a tangent bundle to define the intrinsic distance between two sequences. Guo *et al.* [24] employ the covariance matrix, which naturally lies on a Riemannian manifold, to describe the optimal flow features. They map the covariance matrix to a vector space using the matrix logarithm, and employ a sparse linear representation to encode the human actions. Zhang *et al.* [25] propose a multi-hypergraph learning algorithm to recognize multi-view 3D objects, where the 3D objects are formulated in a manifold, and the correlation among the objects is estimated by label propagation on the manifold structure. We follow these work by describing each local cuboid as a point in a Grassmann manifold, and developing a tangent Fisher vector computational method to aggregate the resulting set of points to obtain a single point that represents the video sequence.

Previous attempts for aggregating a set of LDS points mostly resort to using the bag-of-words (BOW) framework. To this end, it's necessary to cluster a set of given LDSs to form the codewords. Considering that the LDSs are points in a matrix manifold, the traditional clustering techniques that rely on Euclidean representation can not be applied directly. There are currently three ways to address this issue. The first way is to find a low-dimensional Euclidean embedding of the points by means of dimension reduction on the pairwise distance matrix. Then the traditional Euclidean clustering algorithms such as K-means are applied to find the cluster centers in the Euclidean space. Finally the LDS codewords are chosen as those LDSs whose corresponding low-dimensional representations are closest to the cluster centers in the low-dimensional space [26]. The second way is to employ clustering algorithms that directly work with the pairwise distance matrix. Examples include K-medoid [12] and normalized cuts [21]. The third way is to perform clustering on the matrix manifold by using the Karcher mean to represent the LDS codewords [6]. The Karcher mean is defined as the centroid of a density, and is here computed as the point that minimizes a sum of squares of geodesic distances. An intrinsic K-means clustering algorithm on the manifold is used to achieve unsupervised clustering. Ravichandran *et al.* [12] present a bag-of-systems framework by using LDSs to model the spatio-temporal cuboids, and compare dimension reduction plus K-means and K-medoid methods on the pairwise distance matrix to form the codebook. Turaga *et al.* [21] use normalized cuts to cluster a cascade of LDSs, and quantize the cascade of LDSs into a cascade of codewords to investigate the repetitive patterns. Li *et al.* [27] build Hankel matrices on the dense tracklets extracted from a video sequence, and present a bag-of-Hankelets framework by modeling each cluster by a Gamma distribution.

The Fisher kernel framework proposed by Jaakkola and Haussler [19] has many advantages over the above BOW-like framework. First, it encodes high-order statistics of the descriptors. When the distribution is modeled as a parametric generative model, say a GMM, the Fisher vector describe how the set of descriptors of a video deviates from the underlying distribution. Second, the computational expense for the Fisher vector is much less than for the BOW, because it generally requires a smaller codebook. On the other hand, the Fisher kernel framework can be understood as an extension of the BOW framework. The BOW representation is the frequency histogram over the codebook obtained by counting the number of occurrences of codewords. The Fisher vector representation is the gradient of the sample's likelihood with respect to the parameters of the underlying generative model. The generative model can be understood as a probabilistic codebook. From this point of view, the BOW encodes the zero-order statistics of the sample distribution, while the Fisher vector encodes the high-order statistics. The Fisher kernel framework proves its effectiveness over the BOW

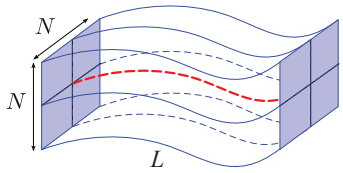


Fig. 2. Illustration of a interest points trajectory and its associated curved spatio-temporal cuboid. Each cuboid is $N \times N$ pixels in space and L frames long in time.

framework on both recognition performance and computational expense. Perronnin and Dance [28] apply the Fisher kernel framework to image categorization by modeling the underlying generative model as a GMM. An image is then represented as a gradient vector with respect to the mean and variance of the GMM. Jégou *et al.* [29] develop an non-probabilistic Fisher vector, named VLAD, to represent images for large scale image search. Wang and Schmid [30] use Fisher vectors to encode the visual and motion descriptors for human action recognition. In this paper, the Fisher kernel framework is extended to incorporate the non-Euclidean descriptors obtained from points in a matrix manifold. We propose a tangent Fisher vector representation method on the matrix manifold, and validate its effectiveness on a wide range of human action data sets.

3 Linear Dynamical Model and Hankel Matrix Representation

In this section, it is shown how a spatio-temporal cuboid is modeled as the output sequence of an LDS in the context of column matrix and Hankel matrix representation. It is then explained what the Hankel matrix representation really does with respect to action recognition. Finally, the observability matrix of an LDS is introduced and an efficient algorithm is proposed to compute it.

3.1 Hankel Matrix Representation for an Observation Sequence

Each curved spatio-temporal cuboid is assumed to be the output sequence of an underlying linear dynamical system. Without loss of generality, let the size of each cuboid be $N \times N$ pixels in space and L frames long in time. We use the column matrix

$$Y_{1:L} = [y_1, y_2, \ldots, y_L] \in \mathbb{R}^{p \times L}, \tag{1}$$

where $p = N \times N$, to represent the observation sequence of raw pixel values. Fig. 2 shows how a curved spatio-temporal cuboid is configured in space and time. Considering that the cuboid is generally small in size on both spatial and temporal domains, it can be modeled by an LDS.

Let $A \in \mathbb{R}^{n \times n}$ be the system dynamic matrix, and $C \in \mathbb{R}^{p \times n}$ be the subspace mapping matrix. Here p and n are the dimensions of the observation space and the state space, respectively. Then a linear time-invariant dynamical system, which generates the observation sequence

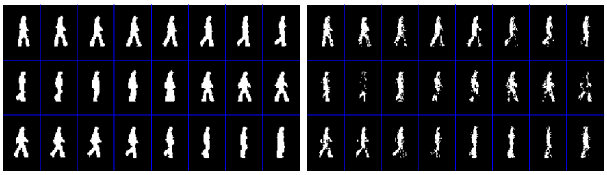


Fig. 3. Original sequence (left) and synthesized sequence (right) generated by an LDS model learned with a walking silhouette sequence from the Weizmann dataset. The model dimension is set to n = 25. The synthesis sequence is generated by setting the initial state equal to the first state of the learned state sequence and evolving 24 steps ahead.

$Y_{1:L}$, is represented by the parameters $\mathbf{M} = (A, C)$, and evolves in time according to the following equations [31]

$$\begin{cases} x_{t+1} = Ax_t + v_t \\ y_t = Cx_t + w_t, \end{cases} \tag{2}$$

where $t = 1, 2, \ldots, L$ is the discrete time index, $x_t \in \mathbb{R}^n$ is the latent state variable, $y_t \in \mathbb{R}^p$ is the vector of pixel values observed at time t, v_t and w_t are the system noise and observation noise, respectively. It is assumed that $v_t \sim \mathcal{N}(0, Q)$ and $w_t \sim \mathcal{N}(0, R)$. Here Q and R are covariance matrices. To learn the parameters of an LDS, a robust algorithm is proposed by [32].

According to the above definition in (2), an LDS can be regarded as a generative model. To show the generation of observations by LDSs, we show in Fig. 3 an original walking sequence and the synthesized one generated by a learned LDS. We first use the raw silhouette data (left) as input to learn the model. Then we use the model parameters to synthesize a new sequence (right) by an evolving process. We see that the synthesized sequence is not only perceptually close to the original one, but also captures the periodic characteristics of walking actions.

In order to explore the observability matrix of an LDS, we reformat the column matrix representation in (1) by introducing a Hankel matrix representation for the observation sequence. Given an observation sequence $Y_{1:L}$, its associated block Hankel matrix $\mathcal{H}^{d,r}(Y_{1:L})$ is a linearly structured matrix of the form [33]

$$\mathcal{H}^{d,r}(Y_{1:L}) = \begin{bmatrix} y_1 & y_2 & \cdots & y_r \\ y_2 & y_3 & \cdots & y_{r+1} \\ \vdots & \vdots & \ddots & \vdots \\ y_d & y_{d+1} & \cdots & y_L \end{bmatrix} \in \mathbb{R}^{pd \times r}, \tag{3}$$

where $d, r \in \{1, 2, \ldots, L\}$ and $r = L - d + 1$.

Each column of the Hankel matrix is a subsequence of the full observation sequence with one time step offset relative to its adjacent columns, and the entries along the block anti-diagonals are the same. The overlapping columns of the Hankel matrix represent the different realizations of the same LDS in response to different initial conditions. The Hankel matrix plays many roles in diverse areas of mathematics, communication and control engineering. Recently, it is also introduced to model the dynamics of visual process, such as dynamic

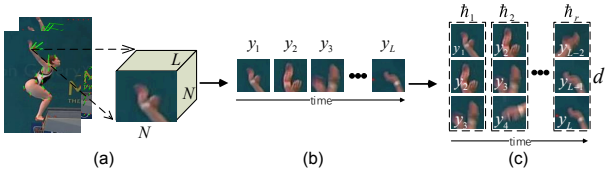


Fig. 4. Illustration of the Hankel matrix representation of an example cuboid extracted from the UCF sports data set. (a) A spatio-temporal cuboid of size $N \times N \times L$. (b) The column matrix representation $Y_{1:L} \in \mathbb{R}^{p \times L}$. Here $p = N \times N$. (c) The Hankel matrix representation $\mathcal{H}^{d,r} \in \mathbb{R}^{pd \times r}$. Here $d = 3$, $r = L - 2$.

texture [34] and human activity [22], [27]. Li *et al.* [27] show that the columns of the Hankel matrix span a subspace that is invariant under changes in the initial conditions and affine transformations of the observation sequence.

According to the realization theory [35], the order or dimension n of an LDS can be expressed as the rank of the associated Hankel matrix, provided that $d, r \geq n$. This suggests a new way to solve the LDS by directly applying stable estimation of the Hankel matrix. In practice, a low order model is generally desired because it is easier and cheaper to analyze. Thus n is typically chosen to be in the range 3-10. The parameters d, r are chosen such that the block Hankel matrix is as near square as possible.

3.2 What Hankel Matrix Does for Action Recognition

Fig. 4 illustrates the reformatting process from the column matrix representation of a cuboid to the Hankel matrix representation for action recognition. The Hankel matrix is constructed by stacking d successive frames for each cuboid frame, till the r frame, where $r = L - d + 1$. With this nonlinear expansion, the Hankel matrix models multiple realizations of the same underlying LDS with different initial conditions. In such a way, the LDS states incorporate information about the future observations. This is particularly helpful for prediction when the motions are periodic. Furthermore, the Hankel matrix implicitly encodes the slowly varying trend and oscillatory nature of the original observation sequence. The slowly varying trend is reported to have much potential for complex action recognition [36].

3.3 Learning the Observability Matrix of an LDS

In Eq. (2), LDS implicitly models the observation sequence $Y_{1:L}$ with a subspace mapping matrix C and a dynamic matrix A. The columns of C describe the principal appearance components, while A represents the state dynamics. The LDS parameters $\mathbf{M} = (A, C)$ are commonly estimated via the singular value decomposition (SVD) to the column matrix $Y_{1:L}$, while constraining the dynamic matrix A to be stable with all its eigenvalues inside the unit circle. The system observability matrix $\mathcal{O}_\infty(\mathbf{M})$ is defined as

$$\mathcal{O}_\infty(\mathbf{M}) = [C^T \quad (CA)^T \quad (CA^2)^T \quad \ldots]^T \in \mathbb{R}^{\infty \times n}. \quad (4)$$

This infinite observability matrix measures how well the hidden states of an LDS can be inferred from its outputs. In practice, a finite observability matrix

$$\mathcal{O}_d(\mathbf{M}) = [C^T \quad (CA)^T \quad \ldots \quad (CA^{d-1})^T]^T \in \mathbb{R}^{pd \times n} \quad (5)$$

is used to approximate the infinite observability matrix $\mathcal{O}_\infty(\mathbf{M})$. The subspace angles between column spaces of the observability matrices are generally used to define the distance [8] for comparing different LDSs.

To learn the finite observability matrix $\mathcal{O}_d(\mathbf{M})$, a straightforward way is to first estimate the model parameters (A, C), and then construct $\mathcal{O}_d(\mathbf{M})$ as defined in (5). However, this approach does not consider the special structure of the Hankel matrix, and thus is not computationally efficient. The observability matrix $\mathcal{O}_d(\mathbf{M})$ is obtained in an efficient way using the Hankel matrix representation.

According to Eq. (2), the expected observation variable satisfies

$$E(y_t) = CA^i x_{t-i}, \quad \forall i \in \{0, 1, \ldots, t-1\}. \quad (6)$$

Substituting (6) into the Hankel matrix (3) yields

$$E(\mathcal{H}^{d,r}) = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{d-1} \end{bmatrix} [x_1 \; x_2 \; \ldots \; x_r] = \mathcal{O}_d(\mathbf{M}) X_{1:r}. \quad (7)$$

This indicates that the expected value of $\mathcal{H}^{d,r}$ can be factorized into an observability matrix and a state sequence. To compute the observability matrix, we decompose the Hankel matrix with the SVD

$$\mathcal{H}^{d,r} = U \Sigma V^T. \quad (8)$$

Then the observability matrix $\mathcal{O}_d(\mathbf{M})$ and the state sequence $X_{1:r}$ are given as

$$\mathcal{O}_d(\mathbf{M}) = U, \quad X_{1:r} = \Sigma V^T. \quad (9)$$

Thus the observability matrix is computed in an efficient way from the Hankel matrix.

Considering the noise assumption in (2), the columns of the orthonormal matrix U span an approximation to the subspace spanned by the columns of the infinite observability matrix. Since this subspace is a point on a Grassmann manifold, the associated LDS can thus be viewed as a point on the Grassmann manifold.

4 Geometry and Statistics on Matrix Manifolds

Given an LDS $\mathbf{M} = (A, C)$ of dimension n and its associated finite observability matrix $\mathcal{O}_d(\mathbf{M})$, the column space of $\mathcal{O}_d(\mathbf{M})$ is an n-dimensional subspace of $\mathbb{R}^{pd}$. The set of all n-dimensional linear subspaces of $\mathbb{R}^m$ is called the Grassmann manifold, and denoted as $\mathcal{G}_{m,n}$. Each point on $\mathcal{G}_{m,n}$ is specified by an $m \times n$ orthonormal matrix, such that the columns of the matrix form an orthonormal basis for the point. In this context, an LDS is represented as a point on the Grassmann manifold $\mathcal{G}_{pd,n}$.

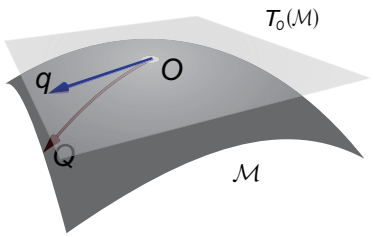


Fig. 5. Illustration of the relation between matrix manifold and tangent space. A point Q on the manifold $\mathcal{M}$ is mapped to a point q on the tangent space $T_O(\mathcal{M})$ at point O.

4.1 Tangent Space of a Matrix Manifold

To measure distances on the Grassmann manifold, we use the fact that the manifold is locally Euclidean and is endowed with a differential structure. This provides a map from a subset of the manifold to the tangent space at a point, and an inverse map from a subset of the tangent space to the manifold. Fig. 5 shows the geometric relation between the matrix manifold and the tangent space.

Formally, let $\mathcal{M}$ be the Grassmann manifold and $O \in \mathcal{M}$ be a point on the manifold. The tangent space at O is denoted as $T_O(\mathcal{M})$. The exponential map $\exp_O : T_O(\mathcal{M}) \to \mathcal{M}$, which maps q in the tangent space $T_O(\mathcal{M})$ onto $\mathcal{M}$, is given by [37]

$$\exp_O(q) = OV \cos\Sigma + U \sin\Sigma, \tag{10}$$

where $U\Sigma V^T$ represents the tangent vector q in the tangent space T_O.

Conversely, the logarithmic map $\log_O : \mathcal{M} \to T_O(\mathcal{M})$, which is the inverse map from $\mathcal{M}$ to $T_O(\mathcal{M})$, is defined in the neighborhood of O by [16], [37]

$$\log_O(Q) = U\Theta V^T, \tag{11}$$

where $U\Sigma V^T = (\mathbf{I} - OO^T)Q(O^TQ)^{-1}$, $\Theta = \arctan(\Sigma)$, and $\mathbf{I}$ is the identity matrix.

Given the above geometry of $\mathcal{M}$, we can study the statistics, such as sample mean and covariance, and derive probability densities, such as GMM, to describe a set of sample points on $\mathcal{M}$.

4.2 Intrinsic GMM Learning Algorithm

In this section, it is shown how the sample mean and covariance are estimated on a Grassmann manifold. An intrinsic GMM learning algorithm is then described.

Let $\{Q_1, Q_2, \ldots, Q_k\}$ be a set of sample points on $\mathcal{M}$. The Karcher mean is defined to be the point that minimizes the sum of squares of geodesic distances

$$\bar{Q} = \arg\min_{Q\in\mathcal{M}} \frac{1}{k}\sum_{i=1}^{k} d(Q, Q_i)^2. \tag{12}$$

To numerically compute the Karcher mean, an iterative algorithm based on the Gauss-Newton gradient descent [38] is commonly used. The iterative updating scheme

Algorithm 1 Intrinsic GMM Learning Algorithm on the Matrix Manifold

Input: Given a set of points $\{Q_1, Q_2, \ldots, Q_N\}$ in the manifold, the number K of Gaussians, and maximum number N_{itr} of iterations;

Initialize K Gaussians $\lambda_j^{(0)} = (\pi_j^{(0)}, \bar{Q}_j^{(0)}, \Sigma_j^{(0)}), j = 1, 2, \ldots, K$ randomly. For $\forall j$, $\pi_j^{(0)} = 1/K$, $\bar{Q}_j^{(0)} \in \{Q_1, Q_2, \ldots, Q_N\}$, $\Sigma_j^{(0)} = \mathbf{I}$. $nItr = 0$;
repeat
  %%% E-step %%%
  for $i = 1 : N, j = 1 : K$ **do**
    Compute the tangent vector $q_{ij} = \log_{\bar{Q}_j^{(nItr)}}(Q_i)$;
    Compute the Mahalanobis distance $d_M(q_{ij}) = (q_{ij}^T {\Sigma_j^{(nItr)}}^{-1} q_{ij})^{1/2}$;
    Make the hard assignment $\gamma_{ij} = 1$ if $j = \arg\min_j d_M(q_{ij})$, and $\gamma_{ij} = 0$ otherwise;
  end for
  %%% M-step %%%
  $nItr = nItr + 1$;
  for $j = 1 : K$ **do**
    Update the Gaussian weight $\pi_j^{(nItr)} = \sum_i \gamma_{ij}/N$;
    Update the Gaussian mean $\bar{Q}_j^{(nItr)}$ using (13);
    Update the Gaussian covariance $\Sigma_j^{(nItr)}$ using (14);
  end for
until Convergence of $\sum_i \sum_j \gamma_{ij} d_M(q_{ij})$, or $nItr > N_{itr}$

Output: $\lambda_j = (\pi_j, \bar{Q}_j, \Sigma_j)$, $j = 1, 2, \ldots, K$.

is given by

$$\bar{Q}^{t+1} = \exp_{\bar{Q}^t}(\epsilon q^t), \quad q^t = \frac{1}{k}\sum_{i=1}^{k} \log_{\bar{Q}^t}(Q_i), \tag{13}$$

where t is the iteration step, and ϵ is the step size which is usually set equal to 0.5.

To define and compute the sample covariance, we use the fact that the tangent space $T_{\bar{Q}}(\mathcal{M})$ at the mean point is a vector space with an inner product. Thus we can use classical statistics in Euclidean space to estimate the covariance. On mapping all the sample points into the tangent space $T_{\bar{Q}}(\mathcal{M})$, the covariance matrix is given as

$$\Sigma = \frac{1}{k-1}\sum_{i=1}^{k} q_i q_i^T, \quad q_i = \log_{\bar{Q}}(Q_i). \tag{14}$$

Once the Karcher mean $\bar{Q}$ and covariance matrix Σ are obtained, a Gaussian distribution is defined on the tangent space by

$$\mathcal{N}(q|\bar{Q}, \Sigma) = \frac{1}{\sqrt{(2\pi)^D|\Sigma|}} \exp\{-\frac{1}{2} d_M^2(q)\}, \tag{15}$$

where D is the dimension of the tangent vector, and $d_M(q) = \sqrt{q^T\Sigma^{-1}q}$ is the Mahalanobis distance of q from the origin of the tangent space.

To learn a K-components GMM for a set of training points $\{Q_1, Q_2, \ldots, Q_N\}$ on the matrix manifold, we seek to estimate K Gaussian clusters with model parameters $(\bar{Q}_j, \Sigma_j)$ and mixture weights π_j for $j = 1, 2, \ldots, K$, so that the weighted sum of squares of Mahalanobis distances is minimized. For this purpose, we propose

an EM-based approach as illustrated in Algorithm 1. We initialize the algorithm by randomly selecting K points as the Gaussian means, and setting all the covariance matrices to be $\mathbf{I}$. In the E-step, we assign each point to the nearest Gaussian with respect to its Mahalanobis distances to the K Gaussian centers. In the M-step, we update each set of Gaussian parameters using the Karcher mean and covariance computation algorithm in (13) and (14).

5 Tangent Fisher Vector

Let $Q = \{Q_i, i = 1, 2, \ldots, s\}$ be the set of manifold points computed from a video sequence. We model the underlying generative model of Q as an intrinsic GMM u_λ. Thus Q can be described by the gradient vector with respect to the parameters λ [29]

$$G_\lambda^Q = \frac{1}{s}\nabla_\lambda \log u_\lambda. \tag{16}$$

The Fisher vector of Q is defined as the normalized gradient vector

$$\mathcal{F}_\lambda^Q = F_\lambda^{-1/2} G_\lambda^Q, \tag{17}$$

where F_λ is the Fisher information matrix of u_λ.

Considering that the intrinsic GMM is defined on a set of tangent spaces centered at the Gaussian means, we call this special Fisher vector the "Tangent Fisher Vector" to differentiate from the traditional Euclidean Fisher vector. Without loss of generality, Let $\lambda_j = (\pi_j, \bar{Q}_j, \Sigma_j), j = 1, 2, \ldots, K$ be the learned GMM on the tangent space at $\bar{Q}_j$. Let q_{ij} be the lift of the point Q_i into the tangent space at $\bar{Q}_j$. The point Q_i is assigned to the Gaussian component for which the Mahalanobis distance is a minimum. Let $\{Q_i, i = 1 : s_j\}$ be the set of points that explain Gaussian λ_j, and let $\{q_{ij}, i = 1 : s_j\}$ be the corresponding tangent vectors. Then for each Gaussian λ_j, we compute the Fisher vector v_j with respect to the mean $\bar{Q}_j$ and covariance Σ_j

$$\begin{aligned} v_{\bar{Q},j} &= \frac{1}{s\sqrt{\pi_j}} \sum_{i=1}^{s} \gamma_{ij} \sigma_j^{-1} q_{ij}, \\ v_{\Sigma,j} &= \frac{1}{s\sqrt{2\pi_j}} \sum_{i=1}^{s} \gamma_{ij} (\sigma_j^{-2} q_{ij}^2 - 1). \end{aligned} \tag{18}$$

where $\sigma_j^2 = diag(\Sigma_j)$ is the diagonal covariance. Considering that the assignment $\gamma_{ij} = 1$ for $j = \arg\min_j d_M(q_{ij})$, and $\gamma_{ij} = 0$ otherwise, we have

$$\begin{aligned} v_{\bar{Q},j} &= \frac{1}{s\sqrt{\pi_j}} \sum_{i=1}^{s_j} \sigma_j^{-1} q_{ij}, \\ v_{\Sigma,j} &= \frac{1}{s\sqrt{2\pi_j}} \sum_{i=1}^{s_j} (\sigma_j^{-2} q_{ij}^2 - 1). \end{aligned} \tag{19}$$

This suggests that in order to compute the tangent Fisher vector v_j, we only need to take care of the points assigned to Gaussian λ_j.

Once we obtain all the $v_j, j = 1 : K$ in all Gaussian components, the tangent Fisher vector of Q is defined as the concatenation of the v_js

$$V = [v_{\bar{Q},1}^T \;\; v_{\Sigma,1}^T \;\; v_{\bar{Q},2}^T \;\; v_{\Sigma,2}^T \;\; \cdots \;\; v_{\bar{Q},K}^T \;\; v_{\Sigma,K}^T]^T. \tag{20}$$

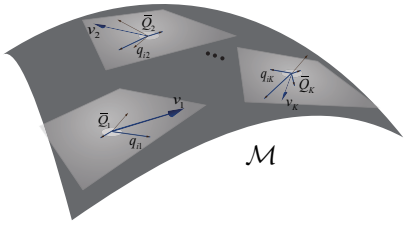


Fig. 6. Illustration of the computation of the tangent Fisher vector. The manifold points obtained from a video sequence are classified using a learned GMM according to the Mahalanobis distance. Then for each Gaussian component, the Fisher vector is computed by aggregating the gradient vectors with respect to the Karcher mean and covariance. Finally, concatenation of the normalized Fisher vectors gives the tangent Fisher vector.

To normalize the tangent Fisher vector V, we apply the intra-normalization [39] followed by the Frobenius norm

$$v_j \leftarrow \frac{v_j}{\sqrt{tr(v_j^T v_j)}}. \tag{21}$$

We show in Fig. 6 the computation of the tangent Fisher vector on the matrix manifold.

Let U and V be the tangent Fisher vectors of two video sequences, respectively. To measure the similarity between them, we define a trace kernel for u_j and v_j based on the inner product on the tangent space

$$\ker(u_j, v_j)_{\mathcal{M}} = tr(u_j^T v_j). \tag{22}$$

The final kernel between U and V is computed as

$$\ker(U, V)_{\mathcal{M}} = \sum_{j=1}^{K} \ker(u_j, v_j)_{\mathcal{M}}. \tag{23}$$

To recognize video sequences, an SVM classifier is firstly trained based on a kernel matrix obtained by computing the pairwise kernels between the TFVs of the training data. A test video sequence is then classified by feeding its TFV to the SVM to yield the class label.

6 Experiments

To evaluate the performance of the proposed method for human action recognition, detailed experiments are carried out on 10 public action data sets, namely UCF sports, IXMAS, Olympic sports, Hollywood2, HMDB51, UCF50, UCF101, THUMOS14, Kinectics400 and ActivityNet v1.2. The experimental setup is introduced and then the results are described and discussed.

6.1 Experimental Setup

6.1.1 Dense Curved Spatio-Temporal Cuboid

To construct the dense curved spatio-temporal cuboids, we use the same parameters as in [18]. Given a video sequence, dense points are first sampled on a grid in each frame using a sampling step size of 5 pixels. Then a dense optical flow field for each frame is computed

w.r.t. its next frame, and dense trajectories are tracked by applying a 3×3 median filter on the optical flow field. The curved spatio-temporal cuboids are constructed as the local space-time volumes along the trajectories, and the size of the cuboid is 32×32 pixels in space and 15 frames long in time.

6.1.2 Observability Matrix of LDS

To learn the observability matrix of an LDS, the observation sequence $Y_{1:L} \in \mathbb{R}^{1024\times 15}$ is first assembled into a Hankel matrix $\mathcal{H}^{d,r}$. Considering the constraint $r = L - d + 1$, and the empirical rule that d, r should have similar values, we set $d = 8, r = 8$ in all the experiments. The model dimension n of the LDS, is set to $n = 3$ by considering the trade-off between recognition performance and computational cost. As a result, the dimensions of the observability matrix are given by $\mathcal{O}_d(\mathbf{M}) \in \mathbb{R}^{8192\times 3}$.

6.1.3 Baseline Descriptors

To quantify the improvement obtained with the LDS, we compare to five baseline descriptors, namely HOG, HOF, MBH, HOG+HOF, and HOG+MBH in both the BOW and FV frameworks. To compute the baseline descriptors, each cuboid is divided into $2 \times 2 \times 3$ cells as in [18]. The final descriptor size is 96 for HOG, 108 for HOF, and 192 for MBH. For the HOG+HOF and HOG+MBH, different descriptors are combined by summing their RBF-χ^2 kernel matrices normalized by their respective mean distance in the BOW framework, or concatenating their normalized Fisher vectors in the FV framework.

6.1.4 Baseline BOW Framework

For the BOW framework, a codebook is trained for each descriptor (HOG, HOF, MBH, and LDS) by randomly selecting a subset of $100,000$ features from the training set, and setting the number of codewords to $4,000$. Each video is represented as a frequency histogram over the codebook for each descriptor. A RBF-χ^2 kernel SVM is used for classification.

6.1.5 FV/TFV Framework

For the FV framework, a GMM is trained for each descriptor (HOG, HOF, and MBH) by randomly selecting a subset of $256,000$ features from the training set, and setting the number of Gaussians to 256. Each video is represented as a Fisher vector over the GMM for each descriptor. A linear SVM is used for classification.

For the TFV framework, an intrinsic GMM is trained by randomly selecting a subset of $256,000$ features from the training set, and setting the number of Gaussians to 256. For each video sequence, each LDS is assigned to one of the Gaussians, and the Fisher vector v_j is computed for each Gaussian. The video sequence is finally represented as a tangent Fisher vector by concatenating all the v_js. A trace kernel SVM is used for classification.

6.1.6 Baseline Clustering Algorithms

To quantify the improvement obtained with the intrinsic GMM in the TFV framework, a comparison is made with four baseline clustering algorithms, namely MDS+K-means [12], K-medoid [12], Normalized cuts [21], and Intrinsic K-means [6] on LDS in both the BOW and TFV frameworks. For the first three methods, as they do not treat the LDS as a point in a Grassmannian, but work directly with the pairwise distance matrices, they are only implemented in the BOW framework. For the last method, as it is designed to work on the Grassmannian, it is compared to both the BOW and TFV frameworks.

6.2 Experimental Results

6.2.1 Evaluation of LDS Parameters

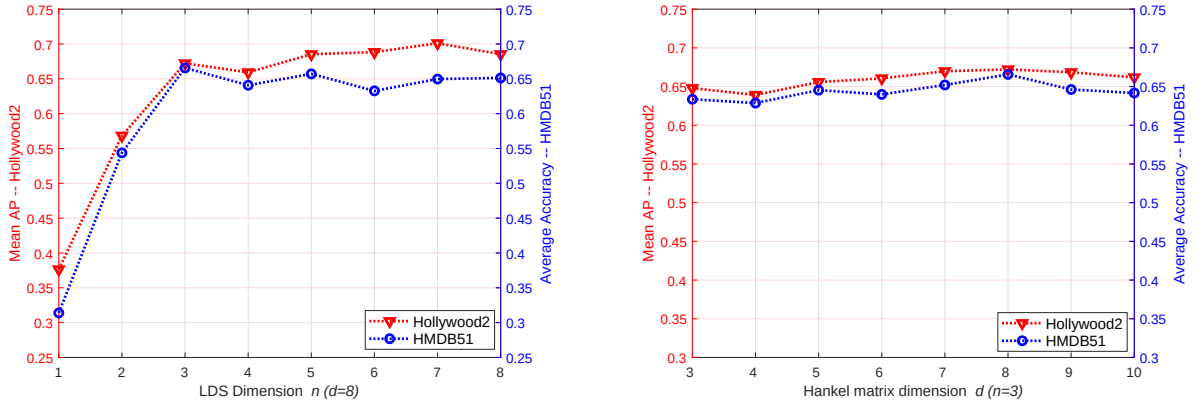
Fig. 7 shows the impact of the LDS parameters in the TFV framework. The mean AP on the Hollywood2 data set and the average accuracy on the HMDB51 data set are reported. These two data sets are chosen for evaluation because they are comparatively more challenging, and the results have large margins for improvement.

The recognition performance with respect to LDS dimension n is shown in Fig. 7a. For both Hollywood2 and HMDB51 data sets, the performance increases rapidly with the increase of n up to $n = 3$. After that, the performance saturates with minor variations around the result obtained at $n = 3$. Though the best results are observed at $n = 7$ for Hollywood2, the computational cost becomes prohibitive. Thus the value $n = 3$ is chosen for all the experiments, in order to save computation time with only a minor loss in the recognition rate.

Fig. 7b shows the recognition results as a function of the Hankel matrix dimension d from $d = 3$ to $d = 10$. Increasing d improves the performance up to $d = 8$. Further increases in d do not yield better results. This is expected in that the block Hankel matrix is square when $d = 8$. Thus $d = 8$ is used as the default parameter in all the experiments unless stated otherwise.

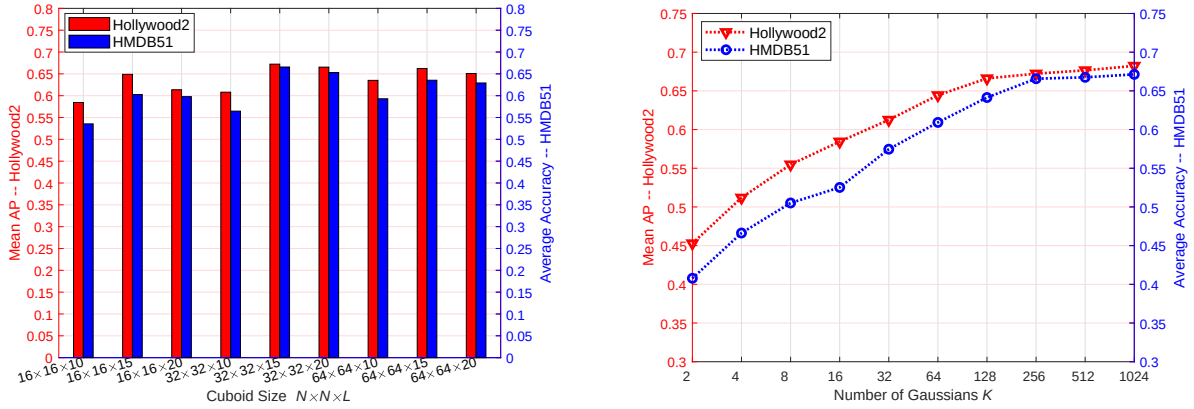
6.2.2 Evaluation of Cuboid Size and the Number of Gaussians

Fig. 8a shows the evaluation results for various sizes of the cuboids in the TFV framework. In particular, nine combinational cuboid sizes obtained from three different spatial sizes ($N = 16, 32, 64$) and three different temporal sizes ($L = 10, 15, 20$) are evaluated on both Hollywood2 and HMDB51 data sets. It is seen that: 1) longer cuboids generally perform better than shorter ones up to $L = 15$. This is because, on the one hand, the LDS needs a minimum length to capture the motion dynamics, on the other hand, long cuboids are very likely to contain some nonlinearities; 2) larger cuboids generally perform better than smaller ones, but the performance saturates after $N = 32$. This is probably because, on the one hand, large cuboids contain more appearance information, on the other hand, oversize cuboids form an over-complete representation of the video sequence. Further increases



(a) (b)

Fig. 7. Evaluation of the LDS parameters on the Hollywood2 and HMDB51 data sets. (a) LDS dimension n, (b) Hankel matrix dimension d.



(a) (b)

Fig. 8. Evaluation of the dense cuboid size and the number of Gaussians in the GMM on the Hollywood2 and HMDB51 data sets. (a) Cuboid size $N \times N \times L$, (b) number K of Gaussians.

in cuboid size do not yield better results. Large cuboids result in the increase in the dimensions of the LDS parameters, and as a result, an increase in the computational cost.

Fig. 8b illustrates the relationship between the recognition performance and the number of Gaussians in the TFV framework. On both Hollywood2 and HMDB51 data sets, the performance increases with the number of Gaussians up to $K = 1024$, but it shows some sign of plateau when the value of K is over 256. Meanwhile, if K is too large, the dimension of the tangent Fisher vector becomes very high, which increases the computational expense dramatically. It is interesting to notice that an mAP of 67.22% is obtained on the Hollywood2 data set and an average accuracy of 66.57% is obtained on the HMDB51 data set at $K = 256$. These results are even better than the ones obtained in the BOW framework with 4000 codewords (60.10% and 47.47% respectively).

6.2.3 Evaluation of Computation Efficiency

In Table 1, we evaluate the computation efficiency of TFV on UCF101 data set with single core CPU. The TFV mainly consists of four modules: constructing dense cuboids, learning observability matrices of LDSs, fitting intrinstic GMM and computing the TFV. We divide the process into two stages, i.e. the first two modules, summarized as LDS feature computation and the last two modules, summarized as TFV feature encoding. We record the test time at different stage with three different video resolutions, because the resolution affects the number of features, which in turn affects the computation time. The classification stage is very fast and its time is negligible.

TABLE 1
Testing Speed of TFV Stages on UCF101 data set (FPS)

	160px	320px	480px
LDS (feature extraction)	18	5	2
TFV (feature encoding)	80	16	8
Performance	82.1%	90.6%	91.5%

We see that as the resolution increases, the FPS drops sharply, but the performance has an obvious improvement. When the resolution is 160px, the TFV can be achieved in real-time, at the cost of moderate performance. In general, LDS is much more time-consuming than TFV, because extracting dense cuboids requires calculation of optical flow, which is very time consuming. To this end, we can use pre-computed cached optical

flow for acceleration.

6.2.4 Comparison to Baseline Descriptors

Table 2 compares the LDS descriptor to five baseline descriptors, namely HOG, HOF, MBH, HOG+HOF, and HOG+MBH in both the BOW and FV/TFV frameworks. The LDS descriptor consistently outperforms all the baseline descriptors on all the data sets. This is mainly because the LDS descriptor intrinsically combines action appearance and dynamics, and thus outperforms HOG, HOF, and MBH, and even their combinations.

HOG is known to capture the static appearance information, and thus gives better results on UCF sports and IXMAS data sets. However, HOG performs poorly on the other data sets, especially on Hollywood2 and HMDB51, because they are recorded in unconstrained environments with large camera motions and cluttered backgrounds.

HOF is designed to encode the motion information, and thus outperforms HOG on nearly all the data sets. A significant improvement of around $4\%-13\%$ is observed on all the data sets in the BOW framework except UCF sports, and an improvement of around $4\% - 8\%$ is observed on all the data sets in the FV framework except IXMAS.

MBH encodes the relative motion over HOF, and thus consistently performs better than HOF on all the data sets. It is observed that MBH gives an average 2% improvement over HOF in both the BOW and FV frameworks, and the improvement is more significant on realistic data sets, e.g., 6% on HMDB51 and UCF50.

HOG+HOF and HOG+MBH slightly improve the performance by around 2% on all the data sets. This is expected, in that the combinations are more discriminative than the individual descriptors, because they capture both appearance and motion information.

LDS consistently outperforms all the baseline descriptors on all the data sets in both the BOW and FV/TFV frameworks. The LDS outperforms all the descriptors by a considerable margin, and improves the best results by around 5%.

In general, the improvement is minor on UCF sports and IXMAS across all the descriptors. This is probably because the performance on these data sets is nearly saturated, and leaves little or no margin for improvement. On the other hand, on realistic data sets, such as Hollywood2 and HMDB51, the improvement is apparent.

6.2.5 Comparison of FV/TFV and BOW Frameworks

The FV/TFV and BOW frameworks on the LDS and baseline descriptors are compared in Table 2. It is seen that the FV/TFV outperforms the BOW on all the descriptors. This is mainly because the FV/TFV encodes the high-order statistics of the descriptors.

For HOG, the improvement varies from 7% to 14%. On UCF50, the result of HOG in the FV framework is even comparable to the result of HOG+MBH in the BOW framework. For HOF, the improvement is trivial on controlled data sets, but significant on realistic data sets, e.g., 10% on Olympic sports and 5% on Hollywood2. The same behavior is also observed for MBH on the data sets. For HOG+HOF and HOG+MBH, the improvement is significant across all the data sets, and reaches 15% on Olympic sports.

For LDS, the improvement is around 5% on all the data sets. On Olympic sports, the improvement is over 8%. It is worth noting that in the TFV framework, LDS achieves results above 90% on nearly all the data sets except the Hollywood2, HMDB51 and THUMOS14. This indicates that the TFV representation of the LDS descriptor is successful over a wide range of action data sets.

6.2.6 Comparison of Different Clustering Algorithms

Table 3 compares different clustering algorithms on the LDS descriptor in both the BOW and TFV frameworks. The intrinsic GMM in the TFV framework achieves the best results on all the data sets.

In general, the performance difference between the baseline algorithms is small in the BOW framework. The MDS+K-means and Normalized cuts perform slightly better than the K-medoid. This is mainly because the former methods consider the geometric property of the pairwise distance matrix, while K-medoid does not do so. However, K-medoid is more computationally efficient than the former ones, because it simply optimizes the cluster centers based on the sum of squared distances. The intrinsic K-means gives the best results in the BOW framework. This is because it computes geometrically accurate cluster centers on the matrix manifold, while the other methods use approximations to the cluster centers given by the points in the training set.

The intrinsic clustering algorithms in the TFV framework significantly outperform the other methods in the BOW framework. The improvement is around 5% on all the data sets. On Olympic sports, the performance gain is over 8%. In addition, the performance difference between the two algorithms is clear. The intrinsic GMM outperforms the intrinsic K-means by around 4% across all the data sets. This is mainly because the intrinsic GMM encodes the high-order statistics of the points on the matrix manifold.

6.2.7 Comparison to Deep Learning Methods

For neural network framework, different backbones, input streams and training strategies will all affect the final performance. Generally speaking, The more layers there are in a neural network, the better performance the deep learning methods will achieve. But at the same time, the training cost and demand for computing resources will also increase. More importantly, compared with the proposed TFV method, deep learning methods are mostly less explanatory. We compare in three aspects the potential of TFV method with recent deep learning methods.

First, TFV performs better than deep learning methods using some backbones. Table 4 compares TFV to the

TABLE 2
Comparison to baseline descriptors on eight data sets in both BOW and FV Frameworks (%)

Data sets	**U.sports**	**IXMAS**	**O.sports**	**Hollywood2**	**HMDB51**	**UCF50**	**UCF101**	**THUMOS14**
Bag-of-words (BOW)								
HOG	84.00	84.79	66.68	39.50	28.89	69.51	67.11	37.57
HOF	82.67	88.24	74.87	52.73	40.44	77.79	74.89	45.23
MBH	85.33	91.70	77.37	54.53	47.30	83.23	81.20	55.24
HOG+HOF	84.00	92.97	78.96	56.92	45.23	82.20	81.98	55.78
HOG+MBH	88.00	92.18	75.24	60.10	47.47	83.76	82.12	62.36
LDS	**90.67**	**93.33**	**83.88**	**61.69**	**58.68**	**87.01**	**85.32**	**62.65**
Fisher vector (FV/TFV)								
HOG	82.67	91.15	76.94	46.41	39.28	83.21	81.55	47.79
HOF	85.33	86.73	84.83	57.45	47.43	85.57	83.26	50.98
MBH	88.00	95.58	89.51	59.47	53.20	88.76	86.78	52.47
HOG+HOF	88.00	95.27	88.02	62.61	52.72	89.69	87.43	61.89
HOG+MBH	89.33	96.61	90.80	63.87	56.45	91.21	89.12	62.56
LDS	**97.33**	**98.79**	**92.67**	**67.22**	**66.57**	**93.58**	**91.55**	**69.89**

TABLE 3
Comparison of different clustering algorithms on LDS descriptor in both BOW and TFV Frameworks (%)

Data sets	**U.sports**	**IXMAS**	**O.sports**	**Hollywood2**	**HMDB51**	**UCF50**	**UCF101**	**THUMOS14**
Bag-of-words (BOW)								
MDS+K-means [12]	89.33	92.97	82.16	59.11	53.90	86.81	84.16	60.25
K-medoid [12]	87.36	91.64	81.42	58.76	52.57	84.98	81.78	57.74
Normalized cuts [21]	90.67	92.18	83.57	60.07	55.22	85.19	83.76	61.28
Intrinsic K-means [6]	90.67	93.35	83.88	61.69	55.68	87.01	86.56	63.56
Tangent Fisher vector (TFV)								
Intrinsic K-means [6]	92.65	96.73	89.92	63.74	58.75	89.88	88.15	65.76
Intrinsic GMM	**97.33**	**98.79**	**92.67**	**67.22**	**66.57**	**93.58**	**91.55**	**69.89**

hidden two-stream CNNs [40] the classification accuracy on four contemporary datasets. We see that when the backbone is temporal segment networks (TSN) [41], the classification results on HMDB51, UCF101, THUMOS14 and ActivityNet1.2 are 66.8%, 93.2%, 74.5% and 87.9% respectively. However when the backbone is VGG16, the experimental results are 60.5%, 90.3%, 66.7% and 77.8%, which are worse than TFV results (66.57%, 91.55%, 69.89% and 79.56%). This is because the network bias is reduced when selecting suitable backbone. Therefore, deep learning method requires comparing different backbones to achieve better results.

Second, different input streams also affect the performance of neural network. In Table 5, we compare the results on the UCF101 dataset of TFV to two-stream CNNs [42] and TSN [41] with different input streams. We see that when RGB or optical flow are used alone, the performance are worse than TFV. Multiple streams fusion helps to achieve better results, but increases the model complexity.

Finally, improper training strategy of neural network will degrade the performance. Table 6 illustrates the results of different training strategies on UCF101 when taking two-stream CNNs [41] as the benchmark. It can be seen that when the neural network does not conduct pre-training or uses some regularization methods, its performance will drop significantly, and even worse than TFV. This indicates that deep learning method has much higher requirements on the training data.

6.2.8 Comparison to State-of-the-art Results

In this section, we compare our proposed method to the state-of-the-art on five challenging action recognition datasets as shown in Table 7. We compare to the most recent related work in the past five years, including both shallow and deep methods. It is observed that our approach has advantages over the shallow methods and some deep methods, but there is still a large margin for improvement compared to the latest deep learning work.

Among the shallow methods, we compare with the most related approaches such as improved dense trajectories [43], stacked Fisher vectors [44] and multi-skip feature stacking [45]. These methods either propose well-designed features or improve the feature coding methods. Basically our method belongs to this category because the proposed TFV is based on the traditional Fisher vector framework. We observe that our method outperform their results on all the datasets.

For the deep methods, we compare to the most recent state-of-the-art approaches, such as AdaScan [46], ActionVLAD [47], Temporal CNN [48], Keyless Attention [49], Hidden Two-Stream [40], MARS [50] and T-SN [41]. Our results are close to some results published before 2018. But after that, the results of deep learning method have developed dramatically both in performance and efficiency.

It is worth noting that our method performs better than the traditional two-stream approach [51], and this approach has inspired many recent deep learning meth-

TABLE 4
Comparison to Hidden Two-Stream CNNs using Different Backbones (%)

	HMDB51	UCF101	THUMOS14	ActivityNet1.2
Hidden two-stream (VGG16) [40]	60.5	90.3	66.7	77.8
Hidden two-stream (TSN) [40]	**66.8**	**93.2**	**74.5**	**87.9**
TFV	66.57	91.55	69.89	79.56

TABLE 5
Comparison to Deep Learning Methods with Different Input Streams on UCF101 (%)

Input	Two-Stream CNNs [42]	TSN [41]	TFV
RGB	84.5	87.7	-
Flow	87.2	-	-
RGB+Flow	92.4	94.9	-
LDS	-	-	91.55

TABLE 6
Comparison to Deep Learning Methods with Different Training Strategies on UCF101 (%)

Training setting	Two-Stream [41]	LDS
From scratch	82.9	-
Pre-train spatial	90.0	-
+Cross modality pre-training	91.5	-
+Partial BN with dropout	92.0	-
TFV	-	91.55

ods for action recognition.

7 Conclusions

LDSs have attracted much research interest in recent years because of their simplicity and efficiency in modelling processes in video sequence. They have the merit of capturing both action appearance and dynamics in an implicit way. In this paper, we further strengthened this merit by encoding the LDS in the tangent Fisher vector framework on a matrix manifold. We introduced a Hankel matrix representation method to assemble the observation data, and proposed an efficient algorithm to simultaneously learn the LDS parameters and observability matrix. We then treated an LDS as a point in a Grassmann manifold, and proposed an intrinsic GMM learning algorithm to cluster the set of LDSs. We finally computed the tangent Fisher vector as the concatenation of the gradient vectors with respect to the GMM centers. We employed the tangent Fisher vector representation of the LDS descriptor on ten public action data sets for human action recognition, and obtained competitive results. In general, the TFV method has a gap in accuracy and efficiency compared to the most recent deep learning methods. The process of LDSs extraction and TFV encoding must be carefully designed, and it is not in an end-to-end manner. In addition, the LDSs only model the local appearance and dynamic information, global information must also be considered. Future work includes few-shot or zero-shot learning against deep learning approaches.

Acknowledgments

This work is supported by the NSFC-general technology collaborative Fund for basic research (Grant No. U1636218), the Natural Science Foundation of China (Grant No. 61751212, 61721004), Beijing Natural Science Foundation (Grant No. L172051), the Key Research Program of Frontier Sciences, CAS, Grant No. QYZDJ-SSW-JSC040, the CAS External cooperation key project, and the National Natural Science Foundation of Guangdong (No. 2018B030311046).

References

[1] J. Aggarwal and M. Ryoo, "Human activity analysis: A review," *ACM Comput. Surv.*, vol. 43, no. 3, pp. 16:1–16:43, 2011.
[2] D. Weinland, R. Ronfard, and E. Boyer, "A survey of vision-based methods for action representation, segmentation and recognition," *Computer Vision and Image Understanding*, vol. 115, no. 2, pp. 224–241, 2011.
[3] A. Bissacco, A. Chiuso, and S. Soatto, "Classification and recognition of dynamical models: The role of phase, independent components, kernels and optimal transport," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 11, pp. 1958–1972, 2007.
[4] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Gaussian process dynamical models for human motion," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, pp. 283–298, 2008.
[5] R. Chaudhry, A. Ravichandran, G. Hager, and R. Vidal, "Histograms of oriented optical flow and Binet-Cauchy kernels on nonlinear dynamical systems for the recognition of human actions," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2009, pp. 1932–1939.
[6] P. Turaga, A. Veeraraghavan, A. Srivastava, and R. Chellappa, "Statistical computations on Grassmann and Stiefel manifolds for image and video-based recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 11, pp. 2273–2286, 2011.
[7] R. J. Martin, "A metric for ARMA processes," *IEEE Trans. Signal Process.*, vol. 48, no. 4, pp. 1164–1170, 2000.
[8] K. De Cock and B. De Moor, "Subspace angles between ARMA models," *Systems and Control Letter*, vol. 46, pp. 265–270, 2002.
[9] S. V. N. Vishwanathan, A. J. Smola, and R. Vidal, "Binet-Cauchy kernels on dynamical systems and its application to the analysis of dynamic scenes," *Int'l J. Computer Vision*, vol. 73, no. 1, pp. 95–119, 2007.
[10] B. Afsari, R. Chaudhry, A. Ravichandran, and R. Vidal, "Group action induced distances for averaging and clustering linear dynamical systems with applications to the analysis of dynamic scenes," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2012, pp. 2208–2215.
[11] G. Doretto, A. Chiuso, Y. N. Wu, and S. Soatto, "Dynamic textures," *Int'l J. Computer Vision*, vol. 51, no. 2, pp. 91–109, 2003.
[12] A. Ravichandran, R. Chaudhry, and R. Vidal, "Categorizing dynamic textures using a bag of dynamical systems," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 2, pp. 342–353, 2013.
[13] P. K. Turaga, A. Veeraraghavan, and R. Chellappa, "From videos to verbs: Mining videos for activities using a cascade of dynamical systems," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2007, pp. 1–8.
[14] G. Luo and W. Hu, "Learning silhouette dynamics for human action recognition," in *Proc. IEEE Conf. Image Processing*, 2013, pp. 2832–2836.

TABLE 7
Comparison of recognition performance to state-of-the-art methods (%)

			HMDB51	UCF101	THUMOS14	Kinectics400	ActivityNet1.2
Stacked Fisher Vectors [44]	2014	shallow	66.8	-	-	-	-
iDT+CNN [TH14 Rank1]	2014	fusion	-	-	71.0	-	-
Two-Stream [51]	2014	deep	59.4	88.0	66.1	61.0	71.9
Multi-skip Feature Stacking [45]	2015	shallow	65.1	89.1	-	-	-
iDT+FV [43]	2016	shallow	60.1	86.0	63.1	-	66.5
AdaScan [46]	2017	deep	66.9	93.2	-	-	-
ActionVLAD [47]	2017	deep	69.8	93.6	-	-	-
Temporal CNN [48]	2018	deep	66.3	92.5	-	-	-
Keyless Attention [49]	2018	deep	-	94.5	-	**77.0**	78.5
Hidden Two-Stream [40]	2018	deep	78.7	97.1	**80.6**	-	**91.2**
MARS [50]	2019	deep	**80.9**	**98.1**	-	74.9	-
TSN [41]	2019	deep	71.0	94.9	80.1	75.7	89.6
TFV+LDS			66.57	91.55	69.89	65.81	79.56

[15] A. Veeraraghavan, A. Srivastava, A. K. Roy-Chowdhury, and R. Chellappa, "Rate-invariant recognition of humans and their activities," *IEEE Trans. Image Process.*, vol. 18, no. 6, pp. 1326–1339, 2009.

[16] Y. M. Lui, "Tangent bundles on special manifolds for action recognition," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 6, pp. 930–942, 2012.

[17] V. Pavlović and J. M. Rehg, "Impact of dynamic model learning on classification of human motion," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2000, pp. 788–795.

[18] H. Wang, A. Kläser, C. Schmid, and C. L. Liu, "Dense trajectories and motion boundary descriptors for action recognition," *Int'l J. Computer Vision*, vol. 103, no. 1, pp. 60–79, 2013.

[19] T. Jaakkola and D. Haussler, "Exploiting generative models in discriminative classifiers," in *Proc. Ann. Conf. Neural Information Processing Systems*, 1998, pp. 487–493.

[20] C. Bregler, "Learning and recognizing human dynamics in video sequences," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1997, pp. 568–574.

[21] P. Turaga, A. Veeraraghavan, and R. Chellappa, "Unsupervised view and rate invariant clustering of video sequences," *Computer Vision and Image Understanding*, vol. 113, no. 3, pp. 353–371, 2009.

[22] B. Li, M. Ayazoglu, T. Mao, O. I. Camps, and M. Sznaier, "Activity recognition using dynamic subspace angles," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2011, pp. 3193–3200.

[23] A. Veeraraghavan, A. Roy-Chowdhury, and R. Chellappa, "Matching shape sequences in video with applications in human movement analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 12, pp. 1896–1909, 2005.

[24] K. Guo, P. Ishwar, and J. Konrad, "Action recognition using sparse representation on covariance manifolds of optical flow," in *Proc. IEEE Int'l Conf. Advanced Video and Signal Based Surveillance*, 2010, pp. 188–195.

[25] Z. Zhang, H. Lin, X. Zhao, R. Ji, and Y. Gao, "Inductive multi-hypergraph learning and its application on view-based 3d object classification," *IEEE Transactions on Image Processing*, vol. 27, no. 12, pp. 5957–5968, 2018.

[26] A. Ravichandran, R. Chaudhry, and R. Vidal, "View-invariant dynamic texture recognition using a bag of dynamical systems," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2009, pp. 1651–1657.

[27] B. Li, O. I. Camps, and M. Sznaier, "Cross-view activity recognition using hankelets," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2012, pp. 1362–1369.

[28] F. Perronnin and C. Dance, "Fisher kernels on visual vocabularies for image categorization," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2007, pp. 1–8.

[29] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Perez, and C. Schmid, "Aggregating local image descriptors into compact codes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 9, pp. 1704–1716, 2012.

[30] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *Proc. IEEE Int'l Conf. Computer Vision*, 2013.

[31] Z. Ghahramani and G. E. Hinton, "Parameter estimation for linear dynamical systems," Dept. Computer Science, Univ. of Toronto, Technical Report CRG-TR-96-2, 1996.

[32] G. Luo, S. Yang, G. Tian, C. Yuan, W. Hu, and S. J. Maybank, "Learning human actions by combining global dynamics and local appearance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 12, pp. 2466–2482, 2014.

[33] P. Van Overschee and B. De Moor, "N4SID: Subspace algorithms for the identification of combined deterministic-stochastic systems," *Automatica*, vol. 30, no. 1, pp. 75–93, 1994.

[34] F. Xiong, O. I. Camps, and M. Sznaier, "Low order dynamics embedding for high dimensional time series," in *Proc. IEEE Int'l Conf. Computer Vision*, 2011, pp. 2368–2374.

[35] M. Moonen, B. D. Moor, L. Vandenberghe, and J. Vandewalle, "On- and off-line identification of linear state space models," *Int'l J. Control*, vol. 49, pp. 219–232, 1989.

[36] Z. Zhang and D. Tao, "Slow feature analysis for human action recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 3, pp. 436–450, 2012.

[37] P.-A. Absil, R. Mahony, and R. Sepulchre, "Riemannian geometry of Grassmann manifolds with a view on algorithmic computation," *Acta Applicandae Mathematicae*, vol. 80, no. 2, pp. 199–220, 2004.

[38] X. Pennec, "Statistical computing on manifolds: From Riemannian geometry to computational anatomy," in *Proc. Emerging Trends in Visual Computing*, 2008, pp. 347–386.

[39] R. Arandjelović and A. Zisserman, "All about VLAD," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2013, pp. 1578–1585.

[40] Y. Zhu, Z. Lan, S. Newsam, and A. Hauptmann, "Hidden two-stream convolutional networks for action recognition," in *Asian Conference on Computer Vision*, 2018, pp. 363–378.

[41] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. V. Gool, "Temporal segment networks for action recognition in videos," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. Early Access, 2019.

[42] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal segment networks: Towards good practices for deep action recognition," in *European Conference on Computer Vision*, 2016, pp. 20–36.

[43] H. Wang, D. Oneata, J. Verbeek, and C. Schmid, "A robust and efficient video representation for action recognition," *Int'l J. Computer Vision*, vol. 119, no. 3, pp. 219–238, 2016.

[44] X. Peng, C. Zou, Y. Qiao, and Q. Peng, "Action recognition with stacked fisher vectors," in *Proc. European Conf. on Computer Vision*, 2014, pp. 581–595.

[45] Z. Lan, M. Lin, X. Li, A. G. Hauptmann, and B. Raj, "Beyond gaussian pyramid: Multi-skip feature stacking for action recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2015, pp. 204–212.

[46] A. Kar, N. Rai, K. Sikka, and G. Sharma, "AdaScan: Adaptive scan pooling in deep convolutional neural networks for human action recognition in videos," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2017.

[47] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell, "ActionVLAD: Learning spatio-temporal aggregation for action classification," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2017.

[48] S. Cho and H. Foroosh, "A temporal sequence learning for

action recognition and prediction," in *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018, pp. 352–361.
[49] X. Long, C. Gan, G. de Melo, X. Liu, Y. Li, F. Li, and S. Wen, "Multimodal keyless attention fusion for video classification," in *AAAI Conference on Artificial Intelligence*, 2018, pp. 7202–7209.
[50] N. Crasto, P. Weinzaepfel, K. Alahari, and C. Schmid, "MARS: Motion-Augmented RGB Stream for Action Recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2019, pp. 1–10.
[51] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in Neural Information Processing Systems 27*, 2014, pp. 568–576.

Guan Luo received the BEng, MEng and PhD degrees in Electronic Engineering from Northwestern Polytechnical University in 1998, 2001 and 2004, respectively. He worked as a Senior Research Associate in RCMT, School of Creative Media, City University of Hong Kong from Jun 2004 to Aug 2005. He is currently an Associate Professor in National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences. His research interests include activity recognition, video analysis, and time-series data mining.

Jiutong Wei received the bachelor's degree in electronic information engineering from Harbin Institute of Technology at Weihai, in 2018. He is currently working toward the Ph.D degree at National Laboratory of Pattern Recognition (NLPR), Institute of Automation, Chinese Academy of Sciences. His research interests include image captioning and video understanding.

Weiming Hu received the PhD degree from Department of Computer Science and Engineering, Zhejiang University in 1998. From April 1998 to March 2000, he was a postdoctoral research fellow with the Institute of Computer Science and Technology, Peking University. Now he is a professor in the Institute of Automation, Chinese Academy of Sciences. His research interests include visual surveillance, and filtering of Internet objectionable information.

Stephen J. Maybank received the BA degree in mathematics from Kings College Cambridge in 1976 and the PhD degree in computer science from Birkbeck College, University of London in 1988. Now he is a professor in the Department of Computer Science and Information Systems, Birkbeck College. His research interests include the geometry of multiple images, camera calibration, visual surveillance, etc.